CLAIMS

We CLAIM:

1. A data structure for scheduling one or more event queues, comprising:

a scheduler having a current time and a current epoch bit;

a first event queue having a first event, the first event including a first time stamp, the first time stamp being associated with a first epoch bit; and

a second event queue having a first event, the first event of the second event queue including a first time stamp, the first time stamp being associated with a first epoch bit; wherein the data structure determines a real time by comparing the current epoch bit with the first epoch bit of the first event queue and the first epoch bit of the second event queue.

2. The data structure of Claim 1, wherein the first time stamp of the first event in the first event queue compares with the first event in the first time stamp of the first event in the second event queue if the first epoch bit associated with the first time stamp of the first event in the first event queue is equal to the first epoch bit associated with the first time stamp of the first event in the second event queue.

3. A method for scheduling one or more event queues and for resolving timestamp rollover conflicts, comprising:

providing a first bit assigning to a scheduler;

providing a second bit that corresponds to an event queue;

comparing the current epoch to determine timestamp rollover conditions by alternately considering the following comparisons of epoch bits to be true when each time counter rolls over.

4. A data structure for scheduling and arbitration, comprising:

in a root node of a heap tree or similar data structure, comprising:

a first sort index for determining priority of a first event; and

a first data field, concatenated to the first sort index, having a first queue identifier associated with the event; and

in a second level, relative to the root node, of the heap tree or similar data structure, comprising:

a second node with a second sort index for determining priority of a second event and a second data field, concatenated to the second sort index, and having a third node with a third sort index for determining priority of a third event; and a third data field, concatenated to the third sort index, having a third queue identifier associated with the third event.

5. A method for strict priority scheduling in a pile, comprising:

in a first event in a first node, comprising:

providing a first queue having a first sort index and a first data field, the first sort index including a first priority and the first data field including a first event queue identifier;

in a second event in a second node, comprising:

providing a second queue having a second sort index and a second data field, the second sort index including a first priority and a second data field including a second event queue identifier;

determining a priority between the first queue and the second queue based on the first priority in the first queue and the second priority in the second queue; and

activating the first queue if the first priority is a higher priority than the second priority and activating the second queue if the second priority is a higher priority than the first priority.

6. The method of Claim 5, wherein the value of the first priority in the first sort index is equal to the value of the first event queue identifier in the first data field.

7. Thee method of Claim 5, wherein the value of the second priority in the second sort index is equal to the value of the second event queue identifier in the second data field.

8. The method of Claim 5, further comprising removing an event from a root node if the first queue in the first event is not empty by rescheduling the event, and percolating a node corresponding to the event down to a location in a heap-like tree structure.

9. The method of Claim 5, further comprising removing an event from a root node if the first queue in the first event is empty by removing the first priority of the first

event in the first node; and leaving an empty node in the first node to percolate down a heap-like structure.

10. The method of Claim 5, further comprising inserting an event in the pile by assigning a designated priority corresponding to a particular queue, and placing the designated priority and a corresponding identifier in a node.

11. A method for ensuring weighted fair queuing in a heap-like tree structure, comprising:

allocating a first service time duration to a first queue having a first event; and

allocating a second service time duration to a second queue having a second event;

if no more events are present in the first queue, comprising:

removing the first queue;

redistributing the first service time duration;

if no more events are present in the second queue, comprising:

removing the second queue;

redistributing the second service time duration.

12. The method of Claim 11, wherein the redistributing of the first service time duration comprises redistributing remaining event queues proportional to a service rate associated with the second queue and other queues excluding the first queue.

13. The method of Claim 11, wherein the redistributing of the second service time duration comprises redistributing remaining event queues proportional to a service rate associated with the first queue and other queues excluding the second queue.

14. The method of Claim 11, further comprising removing an event from a root node if the first queue in the first event is not empty by rescheduling the event, and percolating a node corresponding to the event down to a location in a heap-like tree structure.

15. The method of Claim 11, further comprising removing an event from a root node if the first queue in the first event is empty by removing the first priority of the first event in the first node; and leaving an empty node in the first node to percolate down a heap-like structure.

16. The method of Claim 11, further comprising inserting an event in a pile including:

computing a time required to dispatch an event;

placing a queue identifier of the event, and the time to dispatch the event, in a root node of a heap-like tree structure; and

percolating the node down in the pile.

17. The method of Claim 11, further comprising rescheduling an event queue including:

computing a time required to dispatch a next event in a same queue; and

replacing an old timestamp associated with the event with a new timestamp; and

percolating the node down to a heap-like tree structure.

18. A method for traffic shaping in a pile, comprising:

in a first node, comprising:

a first queue having a first sort index and a first data field, the first sort index having a first priority and a first timestamp, the first timestamp representing a next transmission time for the first queue in the first node, the first data field having a first event queue identifier, the first queue being given a maximum average rate of transmission; and

in a second node, comprising

a second entry having a second sort index and a second data field, the second sort index having a second priority and a second timestamp, the second timestamp representing a next transmission time for the second queue in the second node the second data field having a second event queue identifier, the second queue being given the maximum average rate of transmission.

19. A method for scheduling and arbitrating events, comprising:

in a root node in a heap-like data structure, comprising:

storing an event A in a first event queue having a first priority;

in a second level relative to the root node, comprising:

storing an event C in a second event queue having a second priority; and

dispatching the event A if the first priority is higher priority than the second priority, or dispatching the event C if the second priority is high priority than the first priority .

20. The method of Claim 19, further comprising storing an event B after the event A in the first even queue.

21. The method of Claim 19, further comprising storing an event D, an event E, and an event F in a third event queue.

22. A method for inserting an event, comprising:

providing a node having a first event queue identifier and a first timestamp; and

placing the node at the root node.

23. The method of Claim 22, further comprising percolating the node to a location in a heap-like data structure.

24. A method for removing an event, comprising:

providing a root node having an event including a next event queue;

removing the event from the root node, thereby leaving a hole in the root node; and

percolating the hole to a location in a heap-like data structure.

25. A method for rescheduling an event, comprising:

assigning a first timestamp and a first identifier to a first node;

assigning second time stamp and a second identifier to a second node;

rescheduling the first node by assigning a new timestamp in place of the first timestamp; and

percolating the first node to a location in a heal-like data structure.

26. A data structure system, comprising:

a memory queue, for storing a packet to be transmitted onto an Internet link;

a transmission time calculator, for computing a transmission time of the packet;

a queue parameter table for determining the transmission time on the basis of a specified service rates or hard-coded properties; and

a pile manipulation pipeline, for storing the transmission time and a queue identifier associated with the packet, and for transmitting the packet when sufficient transmission time has elapsed in a root node of a heap like data structure.

27. A multiple piles in a random access memory (RAM), comprising:

in a RAM having a pile data structure, comprising

a first pile, having a first root node representing a first scheduler;

a second pile, linking to the first pile, having a second root node representing a second scheduler.